Density-Aware Sliders

ABSTRACT

We present a new family of selection widgets, or *density-aware sliders*, that utilize the distribution of the underlying data to ease information browsing and selection. Standard sliders suffer from two problems: poor subpixel data querying, and a uniform visual representation of non-uniform data. Our density-aware sliders give users efficient subpixel data querying capabilities using novel graphical elements and embed a visual representation of the data distribution linked to the slider. Through a controlled user study, we find that certain density-aware sliders outperform prior methods for item selection with large datasets.

Author Keywords

Dynamic query, information visualization, sliders, Alphaslider, density-aware slider.

ACM Classification Keywords

H5.2. Information Interfaces and presentation (e.g., HCI): User Interfaces.

INTRODUCTION

Dynamic queries are a powerful and effective interactive technique for filtering information from large datasets [5]. They facilitate rapid, incremental and reversible actions for identifying trends and outliers in data. Several interface widgets make available the advantages of dynamic queries, such as sliders, checkboxes, and buttons.

Sliders are common data filtering widgets. However, standard sliders do not take advantage of the underlying data distribution, and have limited visual feedback of the data density. These problems are especially important when the widget maps more than one item to any pixel along the slider track, also referred to as subpixel mapping or querying facility. Subpixel mapping problems with sliders are evident with the Alphaslider [1], which was designed to query large sorted lists of alphanumeric items (Figure 1-top) by allowing coarse and fine content browsing.

New widgets have been proposed to improve slider interaction [2, 3]. The PVSlider [2] uses a popup vernier to give users sub-pixel-pitch control. The FineSlider [3] uses an elastic band metaphor to give users intuitive control over scrolling speed and precision, where a longer elastic band correlates to faster scrolling and a shorter band correlates to more precise item selection. Zlider [4] uses pressure input to allow the user to fluidly change from coarse to fine item selection. None of the above sliders dynamically adapt to the data they contain. We believe that dynamic adjustment of slider mechanics can improve how these widgets allow for querying through large item sets. We examine this hypothesis by suggesting that the slider's mechanics be adjusted according to the distribution of the data contained in the slider.

We propose *density-aware sliders*, ActiveArea and ActiveList (Figure 1-middle/low), that adjust the size of the slider thumb dynamically based on the items in a category. This allows for non-uniform item browsing based on category density. The density-aware sliders allow for fine control of items with a secondary thumb (triangle). We also introduce different distortion methods as prior methods, such as the Alphaslider, allow categories with more items to consume more physical space on the slider (e.g., letter 'M'). Shrinking the size of the Alphaslider causes categories with fewer items to disappear (e.g., letters 'H', 'I'). We therefore introduce an input distortion that allows all items to have equal footprint on the slider. With input distortion we equip our density-aware sliders with an embedded visualization to show how the underlying data is distributed. Users can modify their filtering behavior by leveraging the density and distribution information provided by our visualizations.

In a comparative evaluation, we find that participants can filter large datasets 15.5% faster with the ActiveList Slider than with the Alphaslider. Our main contribution is in demonstrating the value of new dynamic query sliders based on the distribution of its underlying data set.

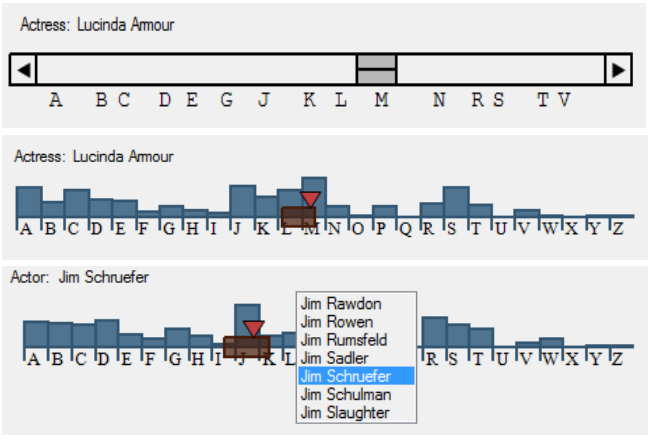

Figure 1: (Top) Display distortion with the Alphaslider. Letter spacing indicates distribution. (Middle) Input distortion with the ActiveArea Slider. Primary slider or thumb is enhanced with a secondary slider (triangle) for fine control. Histograms indicate distribution and user input is distorted based on local density. (Bottom) Input distortion with the ActiveList Slider.

Submitted to ACM CHI 2013.

DESIGN OF MULTI-ITEM PIXEL CONTROL

Mapping multiple items to a single pixel (multi-item pixel mapping) to give users quick access to each item can lead to inefficient exploration of large data sets. The Alphaslider mitigates the multi-item mapping issue through coarse or fine-grained dragging of the slider thumb. Coarse dragging lets the user skip through 10 items at a time while fine-grained dragging lets the user move through the list one item at a time. We call this list-based querying. This can be inefficient in large datasets because of the tremendous amount of cursor movement required to navigate through the list. Furthermore, the Alphaslider's querying method gives no visual feedback when the number of items per pixel exceeds the coarse-grained movement value, making highly dense data even more difficult to navigate.

Subpixel visual feedback is also an issue intricately associated with the Alphaslider. This problem was identified by Ayatsuka et al. [2], who used a popup vernier to provide users with a subpixel visualization. However, their solution lacks the ability to adapt dynamically to the contained data. This requires the application designer to know the exact screen size of the slider as well as how many items it will contain to provide a popup vernier that accurately represents the distribution and density of items.

We propose a multilevel approach that addresses the multi-item pixel mapping problem and provides subpixel visual feedback. A multilevel approach improves subpixel querying by mapping n items in a given category (e.g., letter 'J') to the number of pixels associated with that category (coarse selection). Each pixel in that category contains m items which are mapped across the length of the slider thumb (k to $k + m$). Each pixel of the thumb maps to p items, where $p < m$. Fine control is possible by moving a secondary thumb to query items r to $r + p$ on the thumb (Figure 2).

Our multilevel approach behaves differently depending on the distortion style. *Input distortion* causes the items in any given pixel to vary (m is variable). This means that the number of items mapped along the slider thumb varies, which forces the user to distort their input when they navigate from one category to the next (Figure 1). *Display distortion*, on the other hand, keeps m constant across the entire slider track, allowing users to use uniform input movement for all categories.

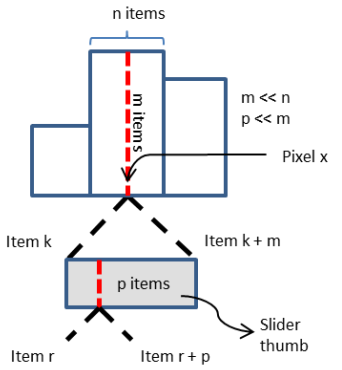

Figure 2: Visual representation of our multilevel approach to subpixel querying. The center histogram contains *n* items. Pixel *x* contains *m* items that are mapped across the slider thumb. Each horizontal pixel of the thumb contains *p* items.

METHODS

We compared different density-aware sliders against the Alphaslider. We did not compare against the FineSlider as this performs suboptimal to the Alphaslider [2].

Apparatus

We built our interfaces using C#.NET. Participants used a standard 3-button mouse with wheel and a 23.5" Dell monitor with a resolution of 1920×1080 pixels. The experimental system consisted of an Intel i5-2400 CPU with a processor speed of 3.1 GHz and 4 GB of RAM. The operating system was Windows 7.

Participants

Twelve participants (20 to 39 years of age) volunteered.

Interfaces

The experiment used three different sliders (Figure 1) and two distortion styles.

Alphaslider

The Alphaslider (Figure 1 - top) provides four navigation capabilities. Users can jump directly to an item in the list by clicking anywhere in the bounded area above the letters and between the arrow buttons. Users can navigate through the list at a rate of ten items per cursor movement by clicking in the top tile of the slider thumb and dragging. Users can also navigate one item at a time either by clicking on the arrows at the ends of the slider or by clicking in the bottom tile of the slider thumb and dragging.

ActiveArea Slider

The ActiveArea Slider (Figure 1 - middle) was inspired by the PVSlider and allows users to navigate by dragging the main rectangular slider or the secondary triangular slider or by rolling the mousewheel. The main slider has a variable width that changes based on the density of the information of the pixel it queries at any given point. The system maps a maximum of four items to each horizontal pixel of the slider. Users can drag the red triangle (a secondary slider) to query items mapped along the width of the main slider. Users can also roll the mouse wheel to navigate one item at a time. These functions provide additional navigation capabilities to the PVSlider. Rolling the mouse wheel moves the secondary slider for continuous querying.

ActiveList Slider

The ActiveList Slider (Figure 1 - bottom) incorporates a list into the ActiveArea Slider. The queried subpixel item determines the contents of the list. The items of the list are the items r through $r + p$ as seen in Figure 2. If p is less than a minimum threshold (six in our design), then the list is enumerated up until that threshold is met or until item $k + m$ is shown. Dragging the secondary slider and rolling the mouse wheel appropriately updates the list. Users cannot drag the secondary slider beyond the left and right edges of the main slider but can use the mousewheel for continuous, uninterrupted querying.

Display Distortion

Display distortion (Figure 1.top) uses variable spacing between letters to represent item distribution. User input is not distorted because item density is uniform across the entire width of the slider. We test this distortion as it was proposed with the Alphaslider [1] but does not display categories with small size.

Input Distortion

Input distortion (Figure 1.middle and bottom) uses histograms to give users information about item distribution. We call this Input Distortion because it causes user input to become distorted based on category density. We propose this distortion to give users visual access to all indices while still providing distribution information.

Hypotheses

The speed with which a user locates an item is largely dependent on total mouse movement. For each of the sliders, there is a period of coarse querying where the user approximates the location of the target, followed by a period of querying where the user performs fine adjustments to acquire the target. Based on this assumption, we provide the following hypotheses:

H1: The ActiveList slider will be the quickest because it simplifies the task optimally by breaking the query into multiple levels.

H2: All of our sliders will outperform the Alphaslider because they give the user better subpixel control and visual feedback.

H3: The ActiveList Slider will be preferred because the technique gives users better, quicker and easier access to the data.

Experimental Design

The experiment used a 3×2×2 within-subject design. The independent variables were:

Technique (*Alphaslider (AS)*, *ActiveArea Slider (AAS)* or *ActiveList Slider (ALS)*)

Distortion Style (*Input Distortion (ID)* or *Display Distortion (DD)*)

Category Density (*Low* or *High*): a category is highly dense if the number of items it contains is greater than or equal to two thirds of the number of items contained in the category with the most items. A category has low density if it contains less than or equal to one third of the number of items contained by the largest category.

We measured target acquisition time and error rate. We also collected participants' preferences and perceived effort and frustration with NASA TLX forms.

Task

The system provided users with a search task for each condition where the user had to find an actor in a database containing 60,000 movies. Each condition had four trials (repetitions). The user had to locate a name from within the 'actor' field in the database. Users started a trial by pressing on a start button and ended it by pressing the spacebar. With each new task, the slider thumb returned to the beginning of its track.

Procedure

Prior to the timed trials, the participants were given a set of training tasks to get familiar with each condition. The actual tasks were ordering according to a balanced Latin square design. Participants filled the NASA TLX (Task Load Index) form for each technique and distortion style. The experimenters also asked participants to write down any comments they had regarding the experiment, techniques, and distortion styles.

RESULTS

We used the univariate ANOVA for analyzing completion time and error rate and the Bonferroni correction for post-hoc pairwise comparison. We used Friedman's X^2 test to analyze the TLX data. Average completion times across techniques, densities and distortion style are summarized in Figure 3.

Acquisition Time: Univariate ANOVA (all assumptions valid) reveals a significant main effect of *technique* ($F_{(2, 22)} = 7.548$, $p = 0.003$) and of *category density* ($F_{(1, 11)} = 5.432$, $p = 0.04$). Lower densities exhibited faster performance times. The ANOVA analysis also reveals significant interaction effect for *category density×distortion style* ($F_{(1, 11)} = 16.672$, $p = 0.02$). We believe this interaction effect arises because the display distortion equalizes the density between categories. Post-hoc pairwise analysis showed significance between the ALS and AS techniques ($p < 0.001$) and between ALS and AAS ($p = 0.001$), with the ActiveArea Slider performing over 15% faster than the AlphaSlider. This supports H1 but not H2.

Error Rate: We found no main effect of error rate on *technique* ($F_{(2, 22)} = 1.000$, $p = 0.384$) or on *input distortion* ($F_{(1, 11)} = 0.000$, $p = 1.000$). Participants were 98% accurate with the AS and 99% accurate with the AAS and the ALS. Participants were also 99% accurate with both distortion styles and both densities.

TLX Analysis: Analysis of the TLX information indicates that there were significant main effects for frustration and effort among interfaces (χ^2 (2) = 7.386, $p < 0.05$ and χ^2 (1) = 18.286, $p < 0.001$, respectively). No significant main effects were found for distortion style for either frustration or effort (χ^2 (2) = 0.034, $p = 0.853$ and χ^2 (1) = 0.533, $p = 0.465$, respectively). Significant interaction effects were found for effort (χ^2 (5) = 22.134, $p < 0.001$) and for frustration (χ^2 (5) = 12.481, $p < 0.03$). Post-hoc analysis with Wilcoxon Signed-Rank Tests and Bonferroni corrections revealed statistically significant differences in effort between the ActiveList Slider with both distortions and every other condition ($p < 0.01$). The post-hoc analysis also revealed no significant differences for frustration. This supports H3.

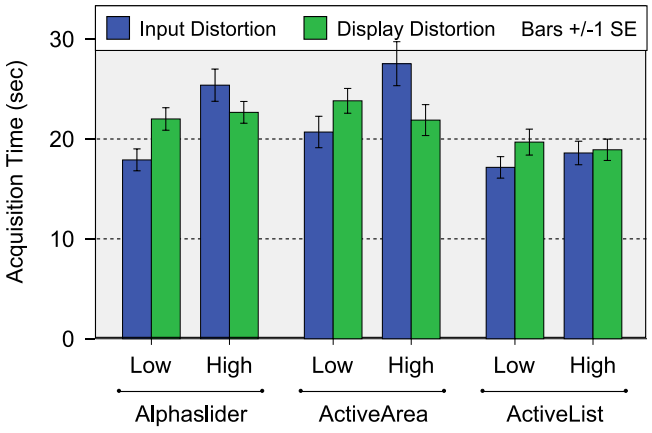

Figure 3: Average completion time across all conditions

DISCUSSION

Our results reveal differences between the ActiveList Slider and the other interfaces. This was primarily a result of the multi-level mapping that allows the thumb slider to change size dynamically and to the presence of the list of items that map to a given pixel on the secondary slider thumb. Our results do not show significant differences between distortion styles. However the ActiveList Slider, performs equally well across both distortion methods.

Overall, participants preferred our sliders to the Alphaslider. This is because the Alphaslider requires users to have fine motor skills in querying one item at a time or to click on arrow buttons repeatedly. Users also commented enjoying the lists provided with the ActiveList Slider.

Lessons Learned

We learned that while users prefer using the input distortion style, they did not perform better with it. We attribute this preference to not requiring non-uniform input offsets and the visual advantage provided by the input distortion over the display distortion. We also learned that users heavily prefer using a mousewheel for fine-grained querying over cursor movement.

Limitations

We suspect that with ultra-dense datasets (greater than 200,000 items) we will start seeing performance degradation with our sliders because of the natural behavior of users. Ideally, users should use the different granularities provided by our sliders: the main thumb, the secondary thumb, and the mousewheel. Users tend not to use the secondary thumb because they perceive it as having the same or similar granularity as the mousewheel, which is true for small to medium density datasets but not for larger datasets. At approximately 20,000 items, the granularity difference between the mousewheel and secondary thumb starts becoming noticeable. We believe that explicit training or a new visual representation of density can solve this concern.

One limitation of the ActiveList Slider is in the amount of space consumed by the list. Our implementation makes the list disappear after the user stops interacting with the slider. Users however found this feature distracting. We intend on investigating other methods for in/revoking the list.

Applications

Aside from using slider in dynamic queries, as we demonstrated in our application, one major application of our work can be in video editing where editors need to select from tens or hundreds of thousands of frames to edit. Our sliders can give editors highly precise frame selection while still adapting to movies of varying length or movies with variable frame rates. Another possible application lies in Maths with precise visual integration. Our sliders can allow users to set upper and lower bounds for visual integration while also giving users the flexibility to adjust the precision on either end of the integral independently while receiving appropriate visual feedback.

CONCLUSIONS AND FUTURE WORK

We have presented density-aware sliders and a new slider distortion style that were designed to improve querying performance in large datasets across multiple densities. We found that our new distortion style was only effective for the ActiveList Slider, allowing this technique to be used in size constrained interfaces. We plan to explore various other conditions under which our density-aware sliders excel. We would also like to investigate other possibilities for density-aware widgets such as a range slider or lasso selection tool.

REFERENCES

1. Ahlberg, C. and Shneiderman, B. (1994) The Alphaslider: A Compact and Rapid Selector. In Proceedings of CHI, 365-371.

2. Ayatsuka, Y., Rekimoto, J., and Matsuoka, S. (1998). Popup Vernier: a Tool for Sub-pixel-pitch Dragging with Smooth Mode Transition. In Proceedings of UIST, 39-48.

3. Masui, T., Kashiwagi, K., and Borden, G.R. IV (1995). Elastic Graphical Interfaces to Precise Data Manipulation. In Proceedings of CHI, 143-144.

4. Ramos, G. and Balakrishnan, R. (2005). Zliding: Fluid Zooming and Sliding for High Precision Parameter Manipulation. In Proceedings of UIST, 143-152.

5. Shneiderman, B. (1998). Dynamic Queries for Visual Information Seeking. IEEE Software, 11:6, 70-77.